

AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently amended) A system for converting a synchronous method call on a target method to an asynchronous method call, the system comprising:

a pattern generator ~~operable to break~~ that breaks the synchronous method call into one or more constituent parts; and

a pattern data store, ~~operably~~ connected to the pattern generator, the pattern data store ~~adapted to store~~ stores data associated with converting ~~[[a]] the~~ synchronous method call to ~~[[an]] the~~ asynchronous method call,

where the one or more constituent parts comprise at least one of: a begin asynchronous operation method; an end asynchronous operation method; an asynchronous call state object; and an asynchronous call result object,

where the begin asynchronous operation method returns ~~the~~ an asynchronous result object and accepts as inputs at least one of: input parameters presented to the target method; input/output parameters presented to the target method; parameters passed by reference to the target method; ~~the~~ an address of an asynchronous callback routine; and the asynchronous call state object,

where the end asynchronous operation method accepts as inputs at least one of: input/output parameters presented to the target method; output parameters presented to the target method; parameters passed by reference to the target method; and the asynchronous call result object, and

where the asynchronous call result object comprises: a first field that holds information concerning whether the begin asynchronous operation method completed asynchronously; and a second field that holds information concerning whether a server completed processing the target method.

- 2-6. (Cancelled)
7. (Previously presented) The system of claim 1, where the asynchronous call result object provides a waitable object.
8. (Original) The system of claim 7, where the asynchronous call result object implements an interface, the interface comprising at least one of:
an asynchronous call state object get method;
a wait handler object get method;
a synchronous call completed field get method; and
a target method call completed field get method.
9. (Currently amended) The system of claim 8, where the end asynchronous operation method is invoked by one of:
an asynchronous call result poller;
the begin asynchronous operation method;
an asynchronous call result waiter; and
an asynchronous callback routine.
10. (Currently amended) The system of claim 9 where the asynchronous call result poller is ~~operable to poll~~ polls the asynchronous call state object to determine whether the server has completed processing the target method.
11. (Currently amended) The system of claim 9 where the asynchronous call result waiter is ~~operable to~~ periodically ~~sleep~~ sleeps and ~~wakeup~~ wakes up, where during an awakened period the asynchronous call result waiter is operable to determine whether the server has completed the target method by examining the asynchronous call result object.
12. (Original) The system of claim 9 where the asynchronous callback routine is invoked when the server has completed the target method.

13. (Previously presented) The system of claim 1, where the pattern generator can convert method calls associated with at least one of file input/output, stream input/output, socket input/output, networking, remoting channels, proxies, web forms, web services and messaging message queues.
14. (Previously presented) The system of claim 1, where the pattern generator can convert method calls associated with file input/output, stream input/output, socket input/output, networking, remoting channels, proxies, web forms, web services and messaging message queues.
15. (Cancelled)

16. (Currently amended) A system to facilitate making asynchronous calls on a target method, the system comprising:

a synchronous method call code associated with a client caller, the synchronous method call code is broken into one or more constituent parts,

where the one or more constituent parts comprise at least one of: a begin asynchronous operation method; an end asynchronous operation method; an asynchronous call state object; and an asynchronous call result object, and

where the asynchronous call result object comprises: a first field that holds information concerning whether the begin asynchronous operation method completed asynchronously; and a second field that holds information concerning whether a server completed processing the target method;

an asynchronous call initializer ~~adapted to accept~~ that accepts input parameters from the client caller and ~~to forward~~ forwards the input parameters towards the target method, the asynchronous call initializer further ~~adapted to establish~~ establishes a callback routine, where the callback routine can be invoked upon completion of the target method, the asynchronous call initializer further ~~adapted to accept~~ accepts a state object and ~~to populate~~ populates one or more fields in the state object with state values associated with the asynchronous call, the asynchronous call initializer further ~~adapted to return~~ returns a result object to the client caller;

an asynchronous call completer ~~adapted to accept~~ that accepts results generated by the target method and ~~to supply~~ supplies the results to the client caller, the asynchronous call completer further ~~adapted to update~~ updates the state object, the asynchronous call completer further ~~adapted to update~~ updates the result object; and

a state tracker, ~~operable to track~~ that tracks and ~~log~~ logs state related to processing associated with the asynchronous call initializer, the asynchronous call completer and the target method, the state tracker further ~~operable to update~~ updates the state object.

17. (Cancelled)

18. (Currently amended) A computer readable medium storing computer executable instructions that performs a method for converting a code for a synchronous method call on a target method to a code for an asynchronous method call, the method comprising:

receiving [[a]] the code for [[a]] the synchronous method call;

passing the code for the synchronous method call through a call conversion process to produce [[a]] the code for [[an]] the asynchronous method call, where the call conversion process comprises: subdividing the code for the synchronous method call into constituent parts; and creating one or more asynchronous method call code segments corresponding to the constituent parts;

creating an asynchronous call result object to store results associated with the asynchronous method call;

creating an asynchronous call state object to store state information associated with the asynchronous method call;

where the constituent parts comprise at least one of:

a begin operation that will not block due to asynchronous method calling; and

an end operation that will not block due to asynchronous method calling;

where the end operation is invoked by one of:

processing associated with polling a field in the asynchronous call state object;

processing associated with waiting on the asynchronous call result object;

the begin operation; and

an asynchronous callback routine;

where code for synchronous method calls is associated with at least one of file input/output, stream input/output, socket input/output, networking, remoting channels, proxies, web forms, web services and messaging message queues can be converted; and

where the asynchronous call result object comprises: a first field that holds information concerning whether the begin asynchronous operation completed asynchronously; and a second field that holds information concerning whether a server completed processing the target method.

19-23. (Cancelled)

24. (Currently amended) A computer readable medium storing computer executable instructions for a method for converting code for a synchronous method call on a target method to code for an asynchronous method call, the method comprising:

dividing the synchronous method call into at least one of a non-blocking asynchronous begin operation and a non-blocking asynchronous end operation; and

associating a call state object to at least one of the non-blocking asynchronous begin operation and the non-blocking asynchronous end operation;

where ~~the~~ a begin asynchronous operation method returns an asynchronous result object and accepts as inputs at least one of: input parameters presented to the target method; input/output parameters presented to the target method; parameters passed by reference to the target method; ~~the~~ an address of an asynchronous callback routine; and ~~the~~ an asynchronous call state object,

where the end asynchronous operation method accepts as inputs at least one of: input/output parameters presented to the target method; output parameters presented to the target method; parameters passed by reference to the target method; and the asynchronous call result object, and

where the asynchronous call result object comprises: a first field that holds information concerning whether the begin asynchronous operation method completed asynchronously; and a second field that holds information concerning whether a server completed processing the target method.

25. (Currently amended) A computer readable medium storing computer executable instructions that performs a method for facilitating asynchronous method calls on a target method, the method comprising:

providing a synchronous method call code broken into constituent parts;
 receiving a request from a calling client to perform processing associated with beginning an asynchronous call to a target method;
 initializing a state tracking object;
 initializing a result object;
 queuing a call to the target method, where the call is queued in a thread pool;
 returning control and a result object to the calling client;
 receiving a request from the calling client to perform processing associated with ending the asynchronous call to the target method;
 returning control and a result consistent with the result of the target method to the calling client upon completion of the processing associated with ending the asynchronous call to the target method;

where the constituent parts comprise at least one of:

a begin operation that will not block due to the asynchronous method calling call;
 and

an end operation that will not block due to the asynchronous method calling call;
 where the begin ~~asynchronous~~ operation ~~method~~ returns the ~~asynchronous~~ result object and accepts as inputs at least one of: input parameters presented to the target method; input/output parameters presented to the target method; parameters passed by reference to the target method; ~~the~~ an address of an asynchronous callback routine; and ~~the asynchronous~~ a call state object,

where the end ~~asynchronous~~ operation ~~method~~ accepts as inputs at least one of: input/output parameters presented to the target method; output parameters presented to the target method; parameters passed by reference to the target method; and the asynchronous call result object, and

where the ~~asynchronous call~~ result object comprises: a first field that holds information concerning whether the begin ~~asynchronous~~ operation completed asynchronously; and a second field that holds information concerning whether a server completed processing the target method.

26. (Original) The method of claim 25 where the request from the calling client to perform processing associated with ending the asynchronous call to the target method is controlled by one of a polling process and a waiting process.

27. (Currently amended) A computer readable medium storing computer executable instructions ~~operable to perform~~ that performs a method for facilitating asynchronous method calls to a target method, the method comprising:

- examining a synchronous method call code and breaking the code into constituent parts;
- accepting one or more requests from a caller to begin an asynchronous call to a target method;

- initializing at least one of a state tracking object and a result object;

- queuing a call to the target method;

- accepting one or more requests to end the asynchronous call to the target method; ~~and~~

- sending at least one of a result and a result object to the caller;

- where the constituent parts comprise at least one of:

- a begin operation that will not block due to an asynchronous method calling; and

- an end operation that will not block due to the asynchronous method calling;

- where the begin ~~asynchronous~~ operation ~~method~~ returns the ~~asynchronous~~ result object and accepts as inputs at least one of: input parameters presented to the target method; input/output parameters presented to the target method; parameters passed by reference to the target method; ~~the~~ an address of an asynchronous callback routine; and ~~the~~ an asynchronous call state object,

- where the end ~~asynchronous~~ operation ~~method~~ accepts as inputs at least one of: input/output parameters presented to the target method; output parameters presented to the target method; parameters passed by reference to the target method; and ~~the~~ an asynchronous call result object, and

- where the asynchronous call result object comprises: a first field that holds information concerning whether the begin ~~asynchronous~~ operation completed asynchronously; and a second field that holds information concerning whether a server completed processing the target method.

28. (Currently amended) A computer readable medium storing computer executable instructions that performs a method for facilitating asynchronous method calls to a target method, the method comprising:

- providing a synchronous method call code broken into constituent parts;
- receiving a request from a calling client to perform processing associated with beginning an asynchronous call to a target method;
- identifying a target method called by a synchronous method call;
- initializing a state tracking object;
- establishing a callback routine, where the callback routine will be invoked upon notification of the completion of the target method, and where the callback routine will invoke processing associated with ending the asynchronous call to the target method;
- queuing a call to the target method, where the call is queued in a thread pool;
- returning and storing control parameters and a result object to the calling client;
- invoking the callback routine upon receiving notification of the completion of the target method;
- performing processing associated with ending the asynchronous call to the target method;
- and
- returning and storing control parameters and a result object consistent with the result of the target method to the calling client upon completion of the processing associated with asynchronous call ~~of~~ to the target method.

29. (Previously presented) A system for converting a synchronous method call to an asynchronous method call, the system comprising:

- means for accepting instructions to call a target method synchronously;
- means for generating instructions to call the target method asynchronously;
- means for generating an object to store results generated in response to performing the instructions to call the target method asynchronously;
- means for generating an object to store state information associated with performing the instructions to call the target method asynchronously;
- means for identifying the target method called by the synchronous method call;
- means for storing parameters intended for the target method;

means for invoking a callback routine when the target method completes; and
means for returning and storing parameters from the target method.

30. (Currently amended) A data packet ~~adapted to be~~ transmitted between two or more computer processes, the data packet comprising:

a first field ~~operable to store~~ that stores information identifying a target method called by a synchronous method call;

one or more second fields ~~operable to store~~ that store parameters intended for the target method;

a third field ~~operable to store~~ that stores information concerning a callback routine to be invoked when the target method completes; and

one or more fourth fields ~~operable to store~~ that store parameters returned from the target method.